http://algs4.cs.princeton.edu

5.1 STRING SORTS

- *strings in Java*
- *key-indexed counting*
- *LSD radix sort*
- *MSD radix sort*
- *3-way radix quicksort*
- *suffix arrays*

5.1 String Sorts

- *strings in Java*
- *key-indexed counting*
- *LSD radix sort*
- *MSD radix sort*
- *3-way radix quicksort*
- *suffix arrays*

String processing

String. Sekuencë e karaktereve.

Abstrakcion i rëndësishëm.

- Sekuencë e gjeneve
- Procesim i informative
- Sistemet e komunikimit (p.sh., e-mail).
- Programimi (p.sh., Java programet).
- ...

“ The digital information that underlies biochemistry, cell biology, and development can be represented by a simple string of G's, A's, T's and C's. This string is the root data structure of an organism's biology. ” — *M. V. Olson*

Lloji i të dhënave char

C char data type. Zakonisht një integer 8-bit.

- Përkrahë 7-bit ASCII.
- Mund të paraqes më së shumti 256 characters.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SP	!	“	#	$	%	&	‘	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Tabela e shndërrimit nga Hexadecimal në ASCII

disa Unicode karaktere

Java char data type. Një16-bit integer i pacaktuar.

- Përkrahë origjinalin e 16-bit Unicode.
- Përkrahë 21-bit Unicode 3.0

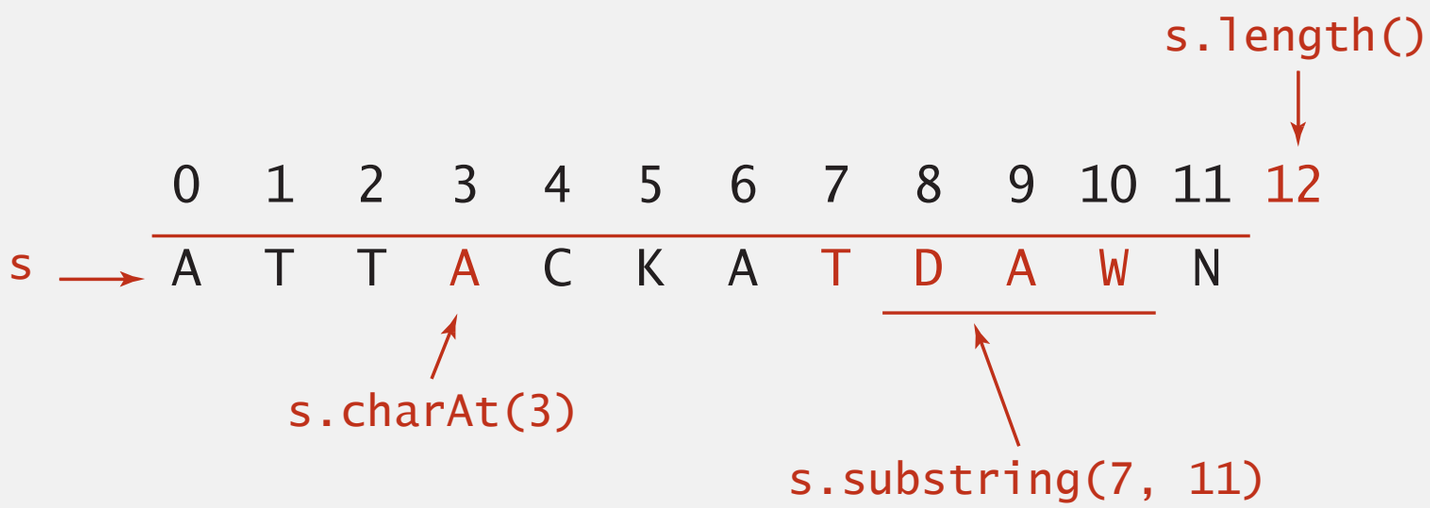
The String data type

String data type in Java. Sekuencë e pandryshueshme e karaktereve

Length. Numri I karaktereve.

Indexing. Get karakterin e $i^{të}$.

Concatenation. Bashkëngjit (lidh) një string me fundin e një stringu tjetër another.



The String data type: paraqitja

Representation (Java 7). Immutable `char[]` array + cache of hash.

operation	Java	running time
length	`s.length()`	1
indexing	`s.charAt(i)`	1
concatenation	`s + t`	$M + N$
⋮		⋮

Krahasimi i dy strings

Sa karaktere krahasohen për të krahasuar dy strings me length W ?

p	r	e	f	e	t	c	h
0	1	2	3	4	5	6	7
p	r	e	f	i	x	e	s

Koha e ekzekutimit. Proporcionale me length të prefiksit më të gjatë të përbashkët

- Proporcionale me W në rastin më të keq.
- Mirëpo, shpeshherë sublinear në W.

Alfabetet

Çelësi digjital Sekuencë e shifrave në një alphabet fiks.

Radix. Numri i shifrave R në një alfabet.

name	R()	lgR()	characters
BINARY	2	1	01
OCTAL	8	3	01234567
DECIMAL	10	4	0123456789
HEXADECIMAL	16	4	0123456789ABCDEF
DNA	4	2	ACTG
LOWERCASE	26	5	abcdefghijklmnopqrstuvwxyz
UPPERCASE	26	5	ABCDEFGHIJKLMNOPQRSTUVWXYZ
PROTEIN	20	5	ACDEFGHIKLMNPQRSTVWY
BASE64	64	6	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdef ghijklmnopqrstuvwxyz0123456789+/
ASCII	128	7	*ASCII characters*
UNICODE16	65536	16	*Unicode characters*

5.1 String Sorts

- strings in Java
- key-indexed counting
- LSD radix sort
- MSD radix sort
- 3-way radix quicksort
- suffix arrays

Pasqyrë e performancës të sorting algoritmeve

Frekuenca e operacioneve.

algorithm	guarantee	random	extra space	stable?	operations on keys
insertion sort	$\frac{1}{2} N^2$	$\frac{1}{4} N^2$	1	✓	`compareTo()`
mergesort	$N \lg N$	$N \lg N$	N	✓	`compareTo()`
quicksort	$1.39 N \lg N$ *	$1.39 N \lg N$	$c \lg N$		`compareTo()`
heapsort	$2 N \lg N$	$2 N \lg N$	1		`compareTo()`

* probabilistic

Kufiri i poshtëm. $\sim N \lg N$ krahasime duhen për cilindo algoritëm që përdorë krahasimet.

Q. A mund të arrihet rezultat më i mirë (me gjithë kufirin)?

A. Po, nëse nuk jemi të varur nga krahasimet me çelësa.

← përdor qasjet me array për të marrë R-way vendime (në vend që të merr vendime binare)

Key-indexed counting: supozime mbi çelësat

Supozim. Çelësat janë integers në mes të 0 dhd $R - 1$.

Implikim. Mund të përdorë çelës si një array index.

Aplikimet.

- Sort string me shkronjën e parë.
- Sort detyrat e punës (klasës) me sekcione.
- Sort numrat e telefonit me kod qyteti.
- Subroutine në një sorting algoritëm.

input		sorted rezultati (me sekcion)	
emri	*sekcion*		
Anderson	2	Harris	1
Brown	3	Martin	1
Davis	3	Moore	1
Garcia	4	Anderson	2
Harris	1	Martinez	2
Jackson	3	Miller	2
Johnson	4	Robinson	2
Jones	3	White	2
Martin	1	Brown	3
Martinez	2	Davis	3
Miller	2	Jackson	3
Moore	1	Jones	3
Robinson	2	Taylor	3
Smith	4	Williams	3
Taylor	3	Garcia	4
Thomas	4	Johnson	4
Thompson	4	Smith	4
White	2	Thomas	4
Williams	3	Thompson	4
Wilson	4	Wilson	4

Çelësat janë integer të vegjël

Key-indexed counting demo

Goal. Sort një array `a[]` me N integers në mes të 0 dhe $R - 1$.

R = 6

- Numëro frekuencën e secilës shkronjë duke përdorur çelësi si index.
- Llogariten frekuencat e mbledhura që specifikojnë destinimet.
- Qaset counteri duke përdor çelësin si index për të move items
- Kopjohet prapa në array origjinale.

```
int N = a.length;
int[] count = new int[R+1];

for (int i = 0; i < N; i++)
   count[a[i]+1]++;

for (int r = 0; r < R; r++)
   count[r+1] += count[r];

for (int i = 0; i < N; i++)
   aux[count[a[i]]++] = a[i];

for (int i = 0; i < N; i++)
   a[i] = aux[i];
```

i	a[i]
0	d
1	a
2	c
3	f
4	f
5	b
6	d
7	b
8	f
9	b
10	e
11	a

a. për 0
b. për 1
c. për 2
d. për 3
e. për 4
f. për 5

Key-indexed counting demo

Goal. Sort një array `a[]` me N integers në mes të 0 dhe $R - 1$.

- Numërohet frekuenca e secilës shkronjë duke përdorur çelës si index.
- Llogariten frekuencat e mbledhura që specifikojnë destinimet.
- Qaset counteri duke përdor çelësin si index për të move items.
- Kopjohet prapa në array origjinale.

numëro frekuencat

```
int N = a.length;
int[] count = new int[R+1];

for (int i = 0; i < N; i++)
   count[a[i]+1]++;

for (int r = 0; r < R; r++)
   count[r+1] += count[r];

for (int i = 0; i < N; i++)
   aux[count[a[i]]++] = a[i];

for (int i = 0; i < N; i++)
   a[i] = aux[i];
```

i	a[i]
0	d
1	a
2	c
3	f
4	f
5	b
6	d
7	b
8	f
9	b
10	e
11	a

offset për 1

r	count[r]
a	0
b	2
c	3
d	1
e	2
f	1
-	3

Key-indexed counting demo

Goal. Sort një array `a[]` me N integers në mes të 0 dhe $R - 1$.

- Numërohet frekuenca e secilës shkronjë duke përdorur çelës si index.
- Llogariten frekuencat e mbledhura që specifikojnë destinimet.
- Qaset counteri duke përdor çelësin si index për të move items.
- Kopjohet prapa në array origjinale.

```
int N = a.length;
int[] count = new int[R+1];

for (int i = 0; i < N; i++)
   count[a[i]+1]++;

for (int r = 0; r < R; r++)
   count[r+1] += count[r];

for (int i = 0; i < N; i++)
   aux[count[a[i]]++] = a[i];

for (int i = 0; i < N; i++)
   a[i] = aux[i];
```

Llogaritja e frekuencave

i	a[i]
0	d
1	a
2	c
3	f
4	f
5	b
6	d
7	b
8	f
9	b
10	e
11	a

r	count[r]
a	0
b	2
c	5
d	6
e	8
f	9
-	12

6 çelësa < d, 8 çelësa < e
prandaj d në a[6] dhe a[7]

Key-indexed counting demo

Goal. Sort një array `a[]` me N integers në mes të 0 dhe $R - 1$.

- Numërohet frekuenca e secilës shkronjë duke përdorur çelës si index.
- Llogariten frekuencat e mbledhura që specifikojnë destinimet.
- Qaset counteri duke përdor çelësin si index për të move items.
- Kopjohet prapa në array origjinale.

```
int N = a.length;
int[] count = new int[R+1];

for (int i = 0; i < N; i++)
   count[a[i]+1]++;

for (int r = 0; r < R; r++)
   count[r+1] += count[r];

for (int i = 0; i < N; i++)
   aux[count[a[i]]++] = a[i];

for (int i = 0; i < N; i++)
   a[i] = aux[i];
```

move items

i	a[i]
0	d
1	a
2	c
3	f
4	f
5	b
6	d
7	b
8	f
9	b
10	e
11	a

r	count[r]
a	2
b	5
c	6
d	8
e	9
f	12
-	12

i	aux[i]
0	a
1	a
2	b
3	b
4	b
5	c
6	d
7	d
8	e
9	f
10	f
11	f

Key-indexed counting demo

Goal. Sort një array `a[]` me N integers në mes të 0 dhe $R - 1$.

- Numërohet frekuenca e secilës shkronjë duke përdorur çelës si index.
- Llogariten frekuencat e mbledhura që specifikojnë destinimet.
- Qaset counteri duke përdor çelësin si index për të move items.
- Kopjohet prapa në array origjinale.

```
int N = a.length;
int[] count = new int[R+1];

for (int i = 0; i < N; i++)
   count[a[i]+1]++;

for (int r = 0; r < R; r++)
   count[r+1] += count[r];

for (int i = 0; i < N; i++)
   aux[count[a[i]]++] = a[i];

for (int i = 0; i < N; i++)
   a[i] = aux[i];
```

copy back →

i	a[i]
0	a
1	a
2	b
3	b
4	b
5	c
6	d
7	d
8	e
9	f
10	f
11	f

r	count[r]
a	2
b	5
c	6
d	8
e	9
f	12
-	12

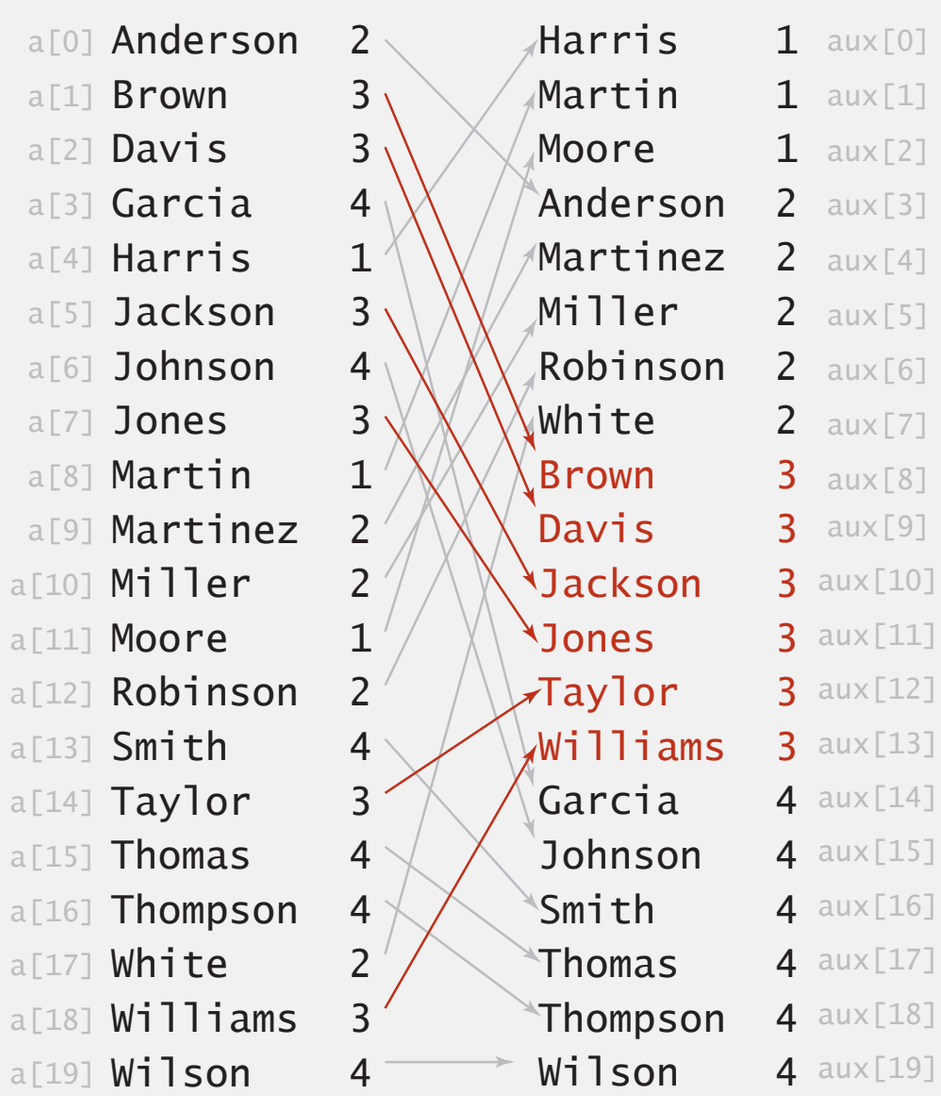
i	aux[i]
0	a
1	a
2	b
3	b
4	b
5	c
6	d
7	d
8	e
9	f
10	f
11	f

Key-indexed counting: analiza

Teoremë. Key-indexed merr kohë proporcionale me $N + R$.

Teoremë. Key-indexed counting zë ekstra hapësirë proporcionale me $N + R$.

Stabile? ✓



5.1 String Sorts

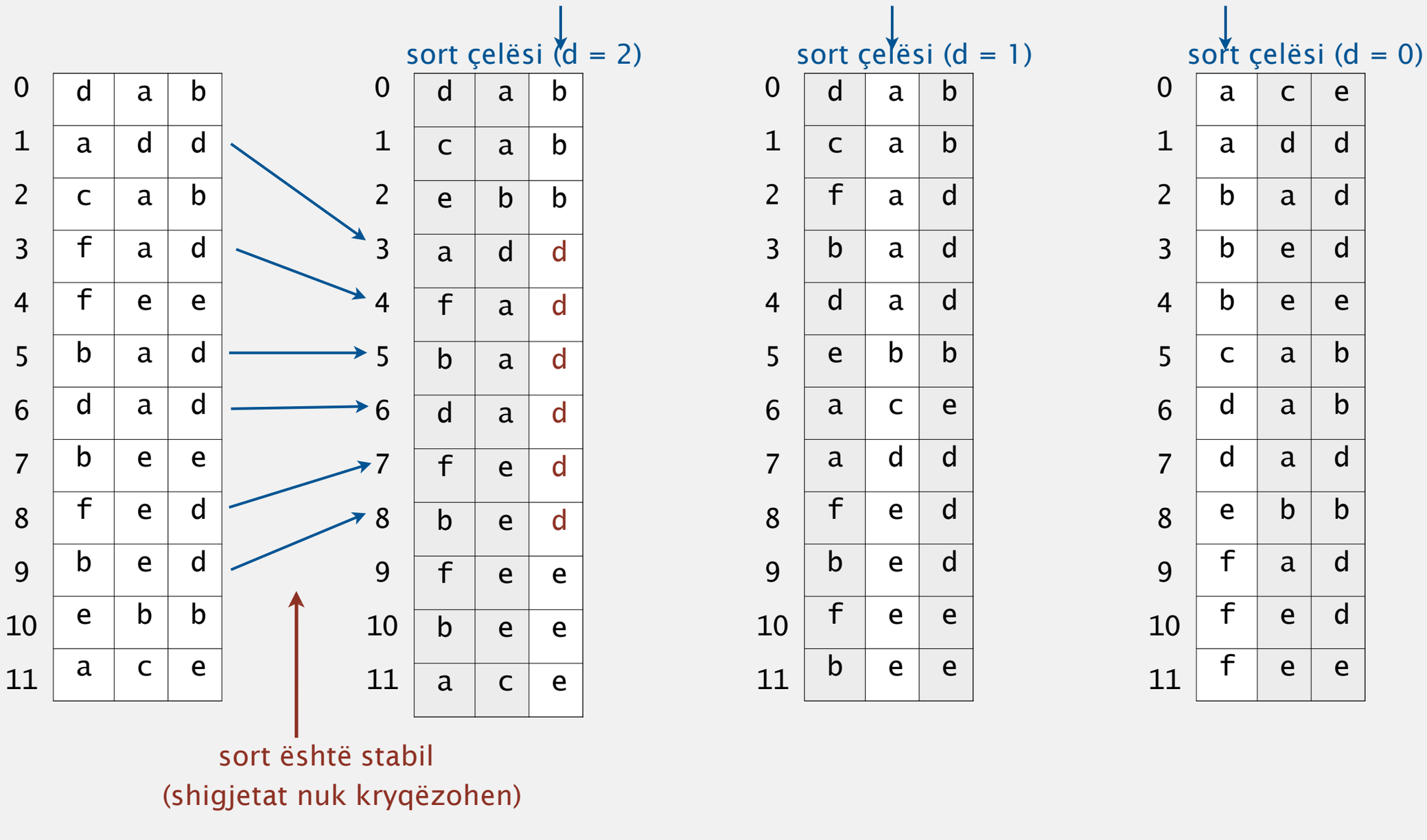
- strings in Java
- key-indexed counting
- LSD radix sort
- MSD radix sort
- 3-way radix quicksort
- suffix arrays

Least Significant Digit string sort

LSD string (radix) sort.

- Të mirren karakteret nga e djathta në të majtë.
- Të sortohen stabil duke përdor karakterin e $d^{të}$ si çelës

(duke zbatuar key-indexed counting).



LSD string sort: Java implementimi

```
public class LSD
{
   public static void sort(String[] a, int W)
   {
      int R = 256;
      int N = a.length;
      String[] aux = new String[N];

      for (int d = W-1; d >= 0; d--)
      {
         int[] count = new int[R+1];
         for (int i = 0; i < N; i++)
            count[a[i].charAt(d) + 1]++;
         for (int r = 0; r < R; r++)
            count[r+1] += count[r];
         for (int i = 0; i < N; i++)
            aux[count[a[i].charAt(d)]++] = a[i];
         for (int i = 0; i < N; i++)
            a[i] = aux[i];
      }
   }
}
```

W strings me length fiks

radix R

key-indexed numërimi
për çdo shifër nga e djathta në të majtë

key-indexed numërimi

Përmbledhje e performancës së sorting algoritmeve

Frekuenca e operacioneve.

algorithm	guarantee	random	extra space	stable?	operations on keys
insertion sort	$\frac{1}{2} N^2$	$\frac{1}{4} N^2$	1	✓	`compareTo()`
mergesort	$N \lg N$	$N \lg N$	N	✓	`compareTo()`
quicksort	$1.39\ N \lg N$ *	$1.39\ N \lg N$	$c \lg N$		`compareTo()`
heapsort	$2\ N \lg N$	$2\ N \lg N$	1		`compareTo()`
LSD sort †	$2\ W\ (N + R)$	$2\ W\ (N + R)$	$N + R$	✓	`charAt()`

* probabilistike

† fixed-length W çelësa

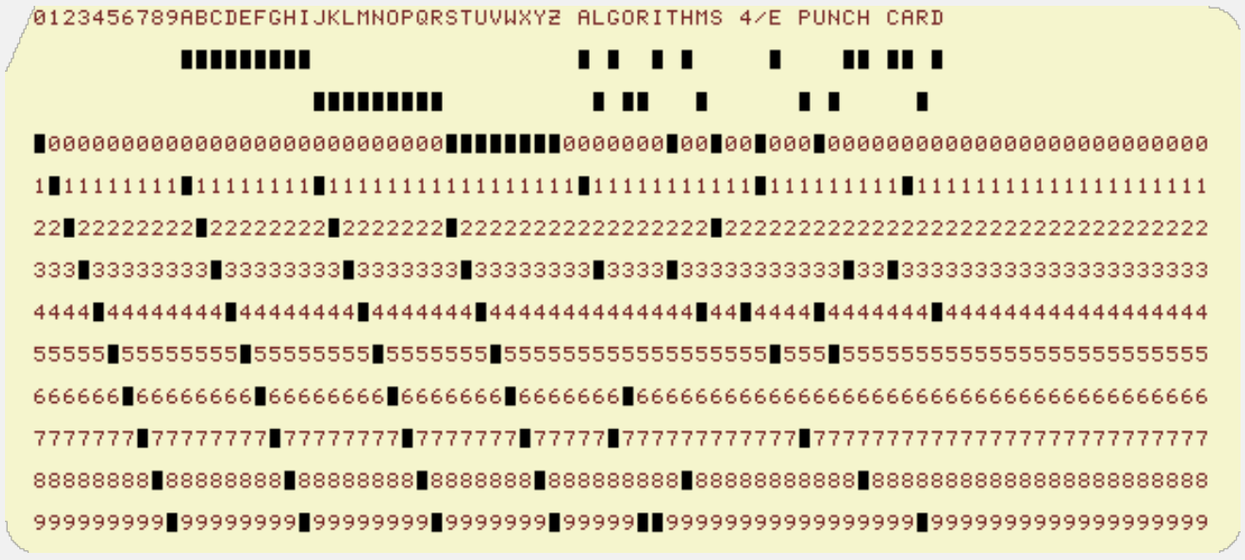
Census në 1900at?

1880 Censusi. 1500 njerëz për 7 vite rradhazi që të përpunohen manualisht të dhënat.

Herman Hollerith. Zhvilloi maqinë automate për numërim dhe sortim

- Përdor punch cards që të regjistrom të dhëna (p.sh., gjinia, mosha).
- Maqina sorton një shtyllat një nga
- Pyetje tipike: sa femra të moshës 20 deri 30 vjeçare?

Maqina dhe sortuesi i Hollerith-it



punch card (12 vrima për shtyllë)

1890 Censusi. Përfundoi në 1 vit (dhe nën buxhet)!

Sorting në 1900at?

Punch cards. [1900at deri 1950at]

- Gjithashtu e dobishme për kontabilitet, regjistrim, dhe biznes procese.
- Medium primar për të dhëna, ruajtje të tyre dhe procesim.

Ndërmarrja e Hollerith-it më vonë u bashkua me tri të tjera për të formuar Computing Tabulating Recording Corporation (CTRC); ndërmarrja u riemërua në 1924.

IBM 80 Series Card Sorter (650 karta për minut)

LSD string sort: një moment historik (1960at)

card punch

punched cards

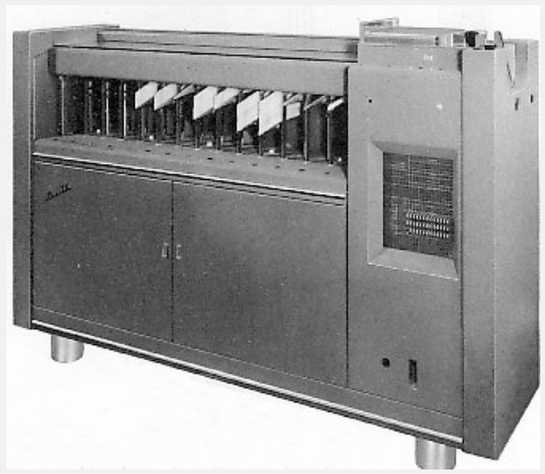
card reader

mainframe

line printer

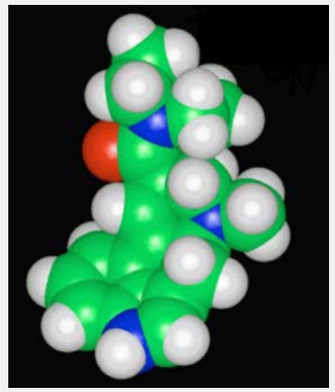
To sort a card deck

- start on right column
- put cards into hopper
- machine distributes into bins
- pick up cards (stable)
- move left one column
- continue until sorted



card sorter

Nuk është drejtpërdrejt
i lidhur me sortim



Lysergic Acid Diethylamide

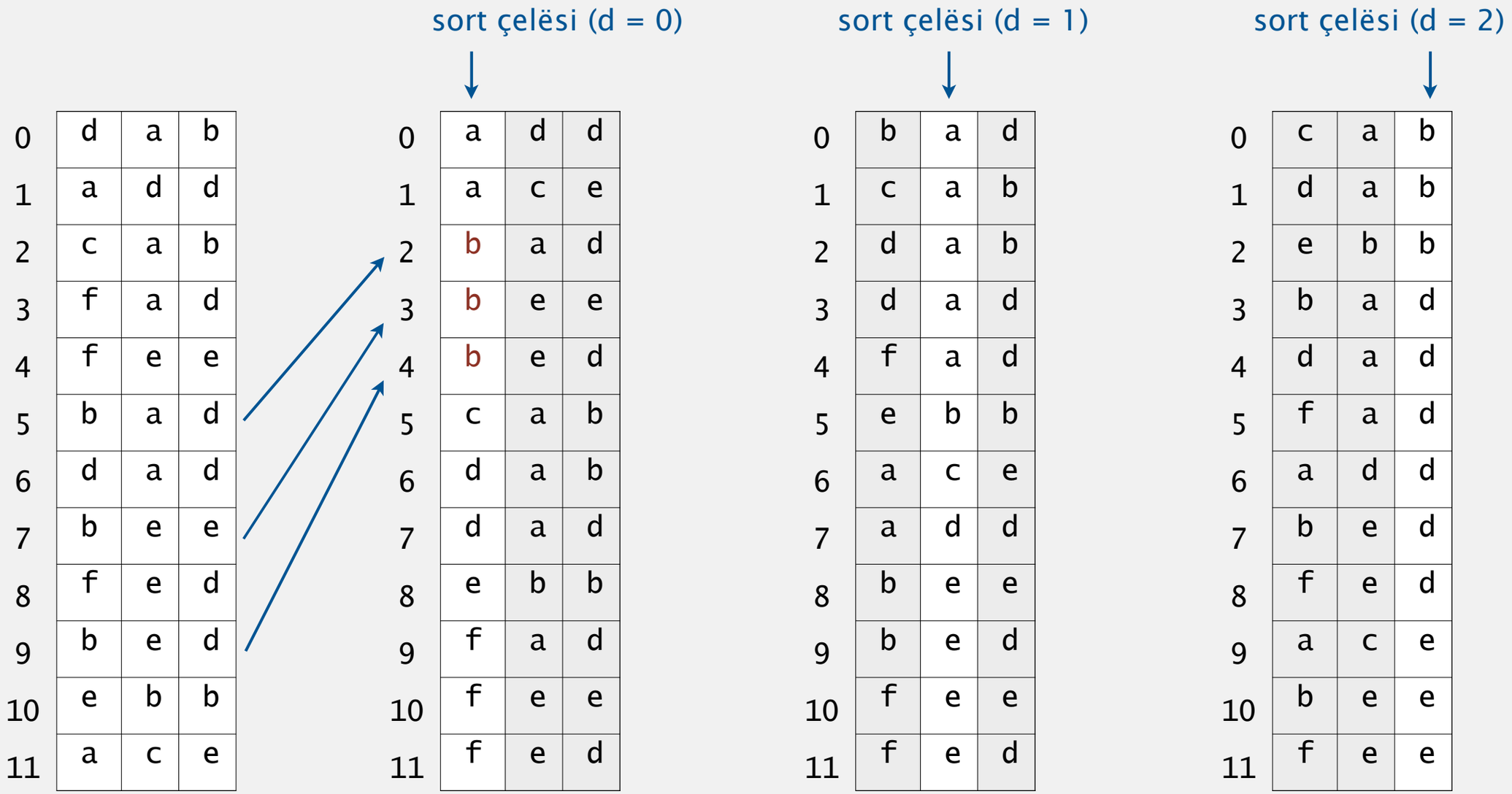
(Lucy in the Sky with Diamonds)

5.1 String Sorts

- strings in Java
- key-indexed counting
- LSD radix sort
- *MSD radix sort*
- 3-way radix quicksort
- suffix arrays

LSD e pasqyruar

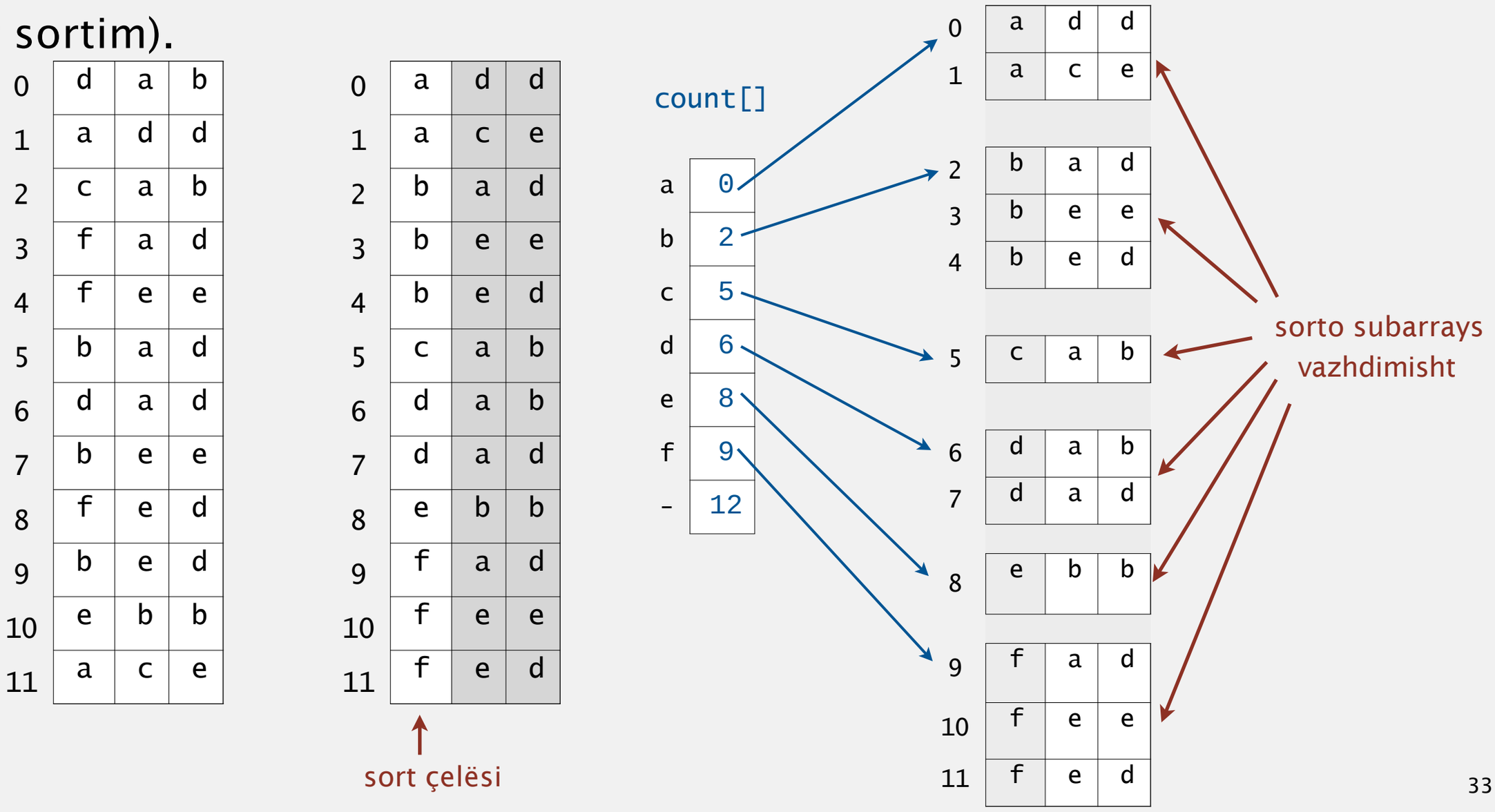
- Të mirren karakteret nga e majta në të djathtë.
- Të sortohet në formë stabile duke përdor karakterin e $d^{të}$ si çelës (duke implementuar key-indexed counting).



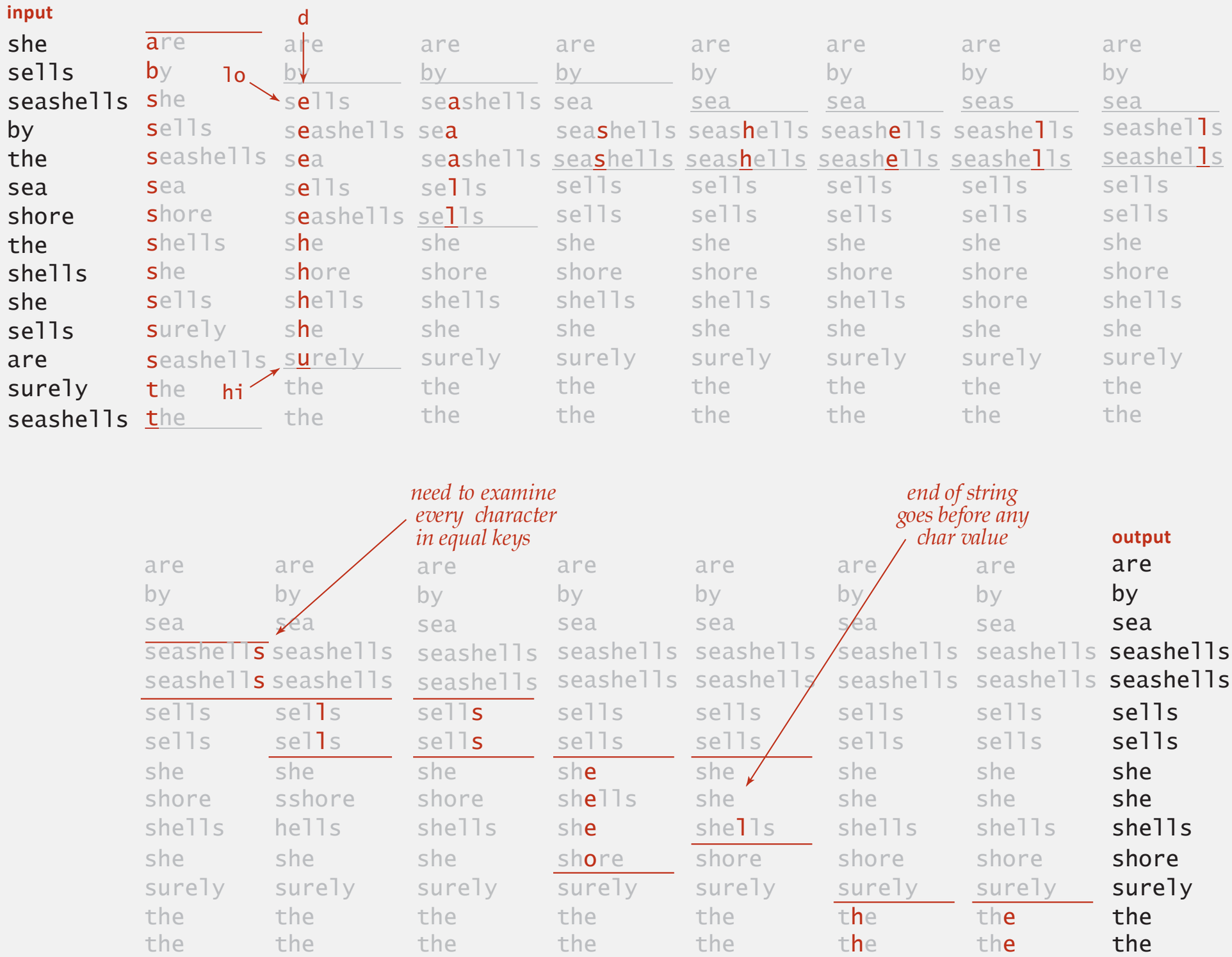
Most-significant-digit-first string sort

MSD string (radix) sort.

- Të ndahet array në R pjes bazuar në karakterin e parë (implemento key-indexed counting).
- Vazhdimisht sorto të gjitha strings që fillojnë me secilin karakter (key-indexed counts përshkruan subarrays për sortim).



MSD string sort: shembull



Trace of recursive calls for MSD string sort (no cutoff for small subarrays, subarrays of size 0 and 1 omitted)

Përmbledhje e performancës së sorting algoritmeve

Frekuenca e operacioneve.

algorithm	guarantee	random	extra space	stable?	operations on keys
insertion sort	$\frac{1}{2} N^2$	$\frac{1}{4} N^2$	1	✓	`compareTo()`
mergesort	$N \lg N$	$N \lg N$	N	✓	`compareTo()`
quicksort	$1.39\ N \lg N$ *	$1.39\ N \lg N$	$c \lg N$		`compareTo()`
heapsort	$2\ N \lg N$	$2\ N \lg N$	1		`compareTo()`
LSD sort †	$2\ W\ (N + R)$	$2\ W\ (N + R)$	$N + R$	✓	`charAt()`
MSD sort ‡	$2\ W\ (N + R)$	$N \log_R N$	$N + D\ R$	✓	`charAt()`

D = function-call stack depth (length e prefiksit më të gjatë të qëlluar)

* probabilistike
† fixed-length W çelësa
‡ average-length W çelësa

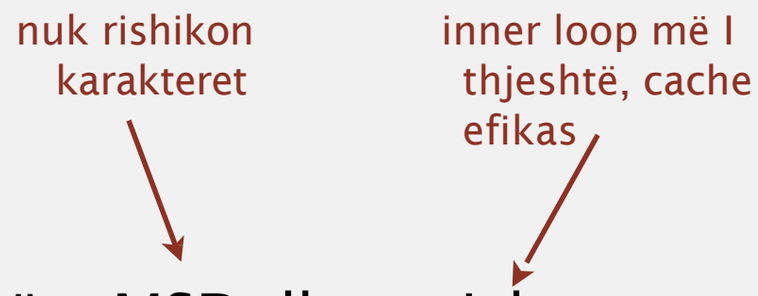
MSD string sort vs. quicksort for strings

Mangësitë e MSD string sort.

- Ekstra hapësirë për `aux[]`.
- Ekstra hapësirë për `count[]`.
- Inner loop ka shumë instrukcione.
- I qaset memories "rastesishëm" (cache-i joefikas).

Mangësitë e quicksort.

- Numër linearitmik të krahasimeve string (jolinear).
- Duhet të rishikon shumë karaktere në çelësat me prefix të gjatë të qëlluar.



Qëllimi Kombino përparësitë e MSD dhe quicksort.

.